Validation of Formal Specification: the Case for Animation

Atif Mashkoor, Faqing Yang, and Jean-Pierre Jacquot

LORIA – Nancy Université
Vandoeuvre-lès-Nancy, France
{firstname.lastname}@loria.fr

Formal methods such as B [1] or Event-B [2] are designed around the idea that a piece of code can be "correct per construction." They use the usual notion of correctness: the program is a mathematically proven implementation of the specification. They are good candidates for industrial use for two major reasons: they embody a development process, refinement, which breaks the notoriously difficult correctness proof into many small and manageable proof obligations, and they have effective tool support.

Through the realization and the analysis of two important case-studies using Event-B in the area of transportation [6,5,8,9], we have discovered that proofs alone are not sufficient to produce a "good" software: we need also to execute, i.e., to test, the software. Two main reasons justify this proposition. Some properties, notably temporal, are *virtually impossible* to model with constructs such as states, invariants, or events which Event-B provides us. We need to ensure that the specification is an adequate model of the problem we want to solve. Software must be verified and validated.

Waiting to have an executable program to begin the validation leads to the same difficulties as proving a program against its specification: costly, very complex, soon unmanageable. The strategy which works with the verification of specifications built by a stepwise refinement process could also work for validation. We should be able to enrich the development process by adding validation activities to each refinement step. Such an enhancement is discussed in [8].

The idea of executing specifications is not new [3]. Some tools have already been developed and integrated into Rodin, ProB [4] or Brama[1], for instance. However, these tools are often unable to animate well written abstract specifications. In fact, qualities sought after for a well written specification, such as abstractness, non determinism, non constructive definitions, are contradictory with what is required for effective computation.

To use validation early in the development cycle, which in our opinion is highly desirable, we need to find techniques to adapt the existing tools. The cost of the technique should be minimal so that it can be frequently used during refinement process.

The first approach to the problem is based on the observation that being provable and being animatable are distinct qualities for specifications: they can be one, the other, or both. It is then possible, in many cases, to "downgrade" a proven, non animatable, specification into a "behaviorally equivalent" animatable, but non provable, specification. In [7], we have proposed several transformations to realize this idea. The validity of such a technique depends on the semantics of the transformations. Since ours are not

[1] http://www.brama.fr

semantic preserving in a strict sense, we had to develop an ad-hoc semantics based on the behavior of the model. The preservation property we want to show can be summed up as: "whatever we observe on the execution of the transformed specification would have been observed on the proven specification." We have obtained some results in this direction.

Despite having transformation rules at our disposal, sometimes animators still fail to execute a specification. For such cases, we expect that translation of Event-B text into an executable programming language can be a reasonable fall-back strategy. We are currently studying this on a 2D platooning model [9] which is out of reach of both ProB and Brama. Current experiments with hand-written systematic translation are promising.

References

1. Abrial, J.R.: The B Book. Cambridge University Press (1996)
2. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press (2010)
3. Balzer, R.M., Goldman, N.M., Wile, D.S.: Operational specification as the basis for rapid prototyping. SIGSOFT Software Engineering Notes 7(5), 3–16 (1982)
4. Leuschel, M., Butler, M.: ProB: An Automated Analysis Toolset for the B Method. Journal on Software Tools for Technology Transfer 10(2), 185–203 (2008)
5. Mashkoor, A., Jacquot, J.P.: Domain Engineering with Event-B: Some Lessons We Learned. In: 18th International Requirements Engineering Conference (RE'10), Sydney, Australia (2010)
6. Mashkoor, A., Jacquot, J.P., Souquières, J.: B événementiel pour la modélisation du domaine: application au transport. In: 9th Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL'09), Toulouse, France (2009)
7. Mashkoor, A., Jacquot, J.P., Souquières, J.: Transformation Heuristics for Formal Requirements Validation by Animation. In: 2nd International Workshop on the Certification of Safety-Critical Software Controlled Systems (SafeCert'09), York, UK (2009)
8. Yang, F., Jacquot, J.P.: Prouvé ? Et après ? In: 10th Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL'10), Poitiers, France (2010)
9. Yang, F., Jacquot, J.P.: Scaling up with Event-B: A Case Study. In: 3rd NASA Formal Methods Symposium (NFM'11), California, USA (2011)